The MINOS BeamData Process

Brett Viren
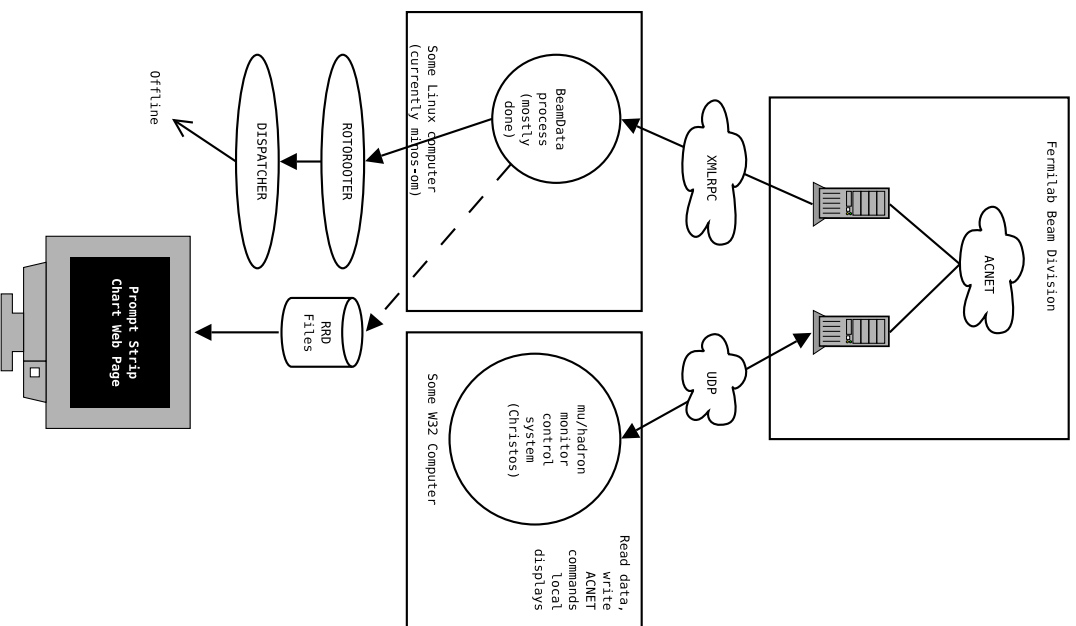
Brookhaven National Lab

bv@bnl.gov

- Beam Monitoring Data Context

- BeamData process details

- How to run the process

- Configuring

- The RawDataBlock offline interface

- Prompt monitoring

- To do list

Fermilab Beam Division

ACNET

XMLRPC

UDP

Some Linux computer
(currently m*os-om)

BeamData
process
(mostly
done)

ROTOROOTER

DISPATCHER

Offline

RRD
Files

Prompt Strip
Chart Web Page

Some W32 Computer

mu/hadron
monitor
control
system
(Christos)

Read data,
write
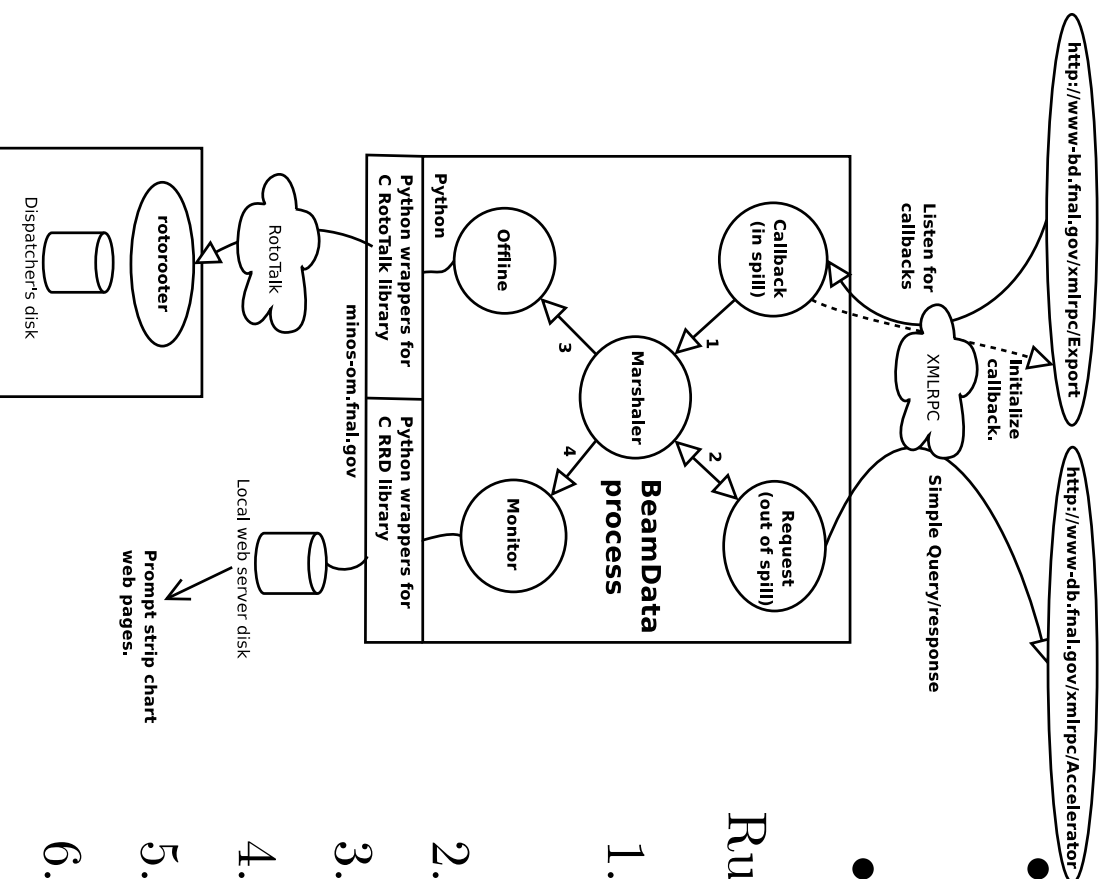ACNET
commands
local
displays

BeamData process

- Access via XMLRPC

- Read only, no control

- Scaled data

- Data path to offline

Control process

- Access via lowlevel UDP

- Read/Control

- Unscaled data (?)

- Data consumed locally

- **Two data sources: in-spill "triggered" callback and out-of-spill query/response**

- **Two data sinks: offline and prompt monitoring**

Run time order

1. Give list of in-spill devices and trigger device to XMLRPC

2. Go in to listen mode

3. Callback or query response arrives

4. If callback, send delayed query for out-of-spill

5. Forward data to sinks

6. go to 2

---

http://www-bd.fnal.gov/xmlrpc/Export

http://www-db.fnal.gov/xmlrpc/Accelerator

Listen for callbacks

Initialize callback.

Simple Query/response

XMLRPC

Callback (in spill)

Request (out of spill)

Marshaler

Monitor

Offline

BeamData process

Python

Python wrappers for C RotoTalk library

Python wrappers for C RRD library

minos-om.fnal.gov

RotoTalk

rotorooter

Dispatcher's disk

Local web server disk

Prompt strip chart web pages.

Currently the idea is to run on minos-om.fnal.gov

As `minos@minos-om.fnal.gov`:

1. `setup_beamdata`

2. `cd BD/srt/BeamData/run`

3. `./start.sh`

Starts local `rotorooter`, `msgReader` and `minosbd.py` (the BeamData process)

Output goes to `~/BD/data/B??????_??????.mbeam.root`.

Current setup is for testing only, needs to be hooked into runcontrol.
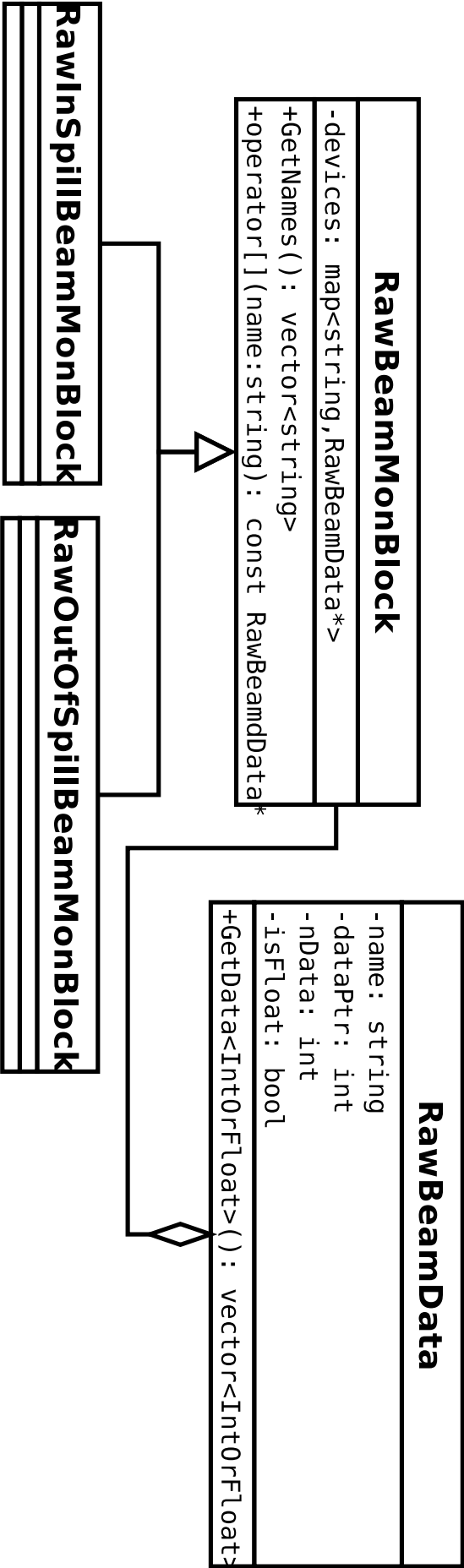
Possible to run from inside a firewall via SSH tunnels.

Configuration is a cascade of:

1. Hard coded defaults in the `config.py` class

2. INI style config file in `~user/.minosbd.cfg`

3. INI style config file in `./minosbd.cfg`

4. Command line arguments

Options:

- In- and out-of-spill as well as trigger devices

- Logging levels and output location

- Connection/callback URL's, hosts, ports

- Rotorooter host, output file sizes

RawBeamMonBlock

-devices: map<string,RawBeamData*>

+GetNames(): vector<string>
+operator[](name:string): const RawBeamdData*

RawInSpillBeamMonBlock

RawOutOfSpillBeamMonBlock

RawBeamData

-name: string
-dataPtr: int
-nData: int
-isFloat: bool

+GetData<IntOrFloat>(): vector<IntOrFloat>

- In- and out-of-spill unpackers implemented in RawBeamMonBlock,

- which acts like a map between ACNET device name and RawBeamData

- which holds the data for one ACNET device

- RawBeamMonData is either int or float as determined by the data

- Failed lookups or type mismatch trigger errors

7/7 – To do

- Hook into run control

- Prompt monitoring web pages

- Runtime safeguards (no duplicate process, etc)

- Direct output ROOT files into Enstore

- Flesh out and test RawDataBlock, particularly w.r.t. time stamping

- Testing, testing, testing...

Will have 2 weeks FC shift in 2nd part of October, will do final work on this then, but much of this should be done before hand.